COMP9321 Data Services Engineering

Semester 2, 2018

Week 3: Data Cleansing and Manipulation

Agenda

- Understanding your data
- Data Cleansing
- Data Manipulation

Understanding the Data (ask the right Questions)

- What is this dataset?
- What should I expect within this dataset?
- Basic concepts (e.g., domain knowledge)
- What are the questions that I need to answer?
- Does the dataset have some sort of a schema? (utilize domain knowledge)

Understanding the Data using Python

- You can use the describe() function to get a summary about the data excluding the NaN values. This function returns the count, mean, standard deviation, minimum and maximum values and the quantiles of the data.
- Use pandas .shape attribute to view the number of samples and features we're dealing with
- it’s also a good idea to take a closer look at the data itself. With the help of the head() and tail() functions of the Pandas library, you can easily check out the first and last 5 lines of your DataFrame, respectively.
- Use pandas .sample attribute to view a random number of samples from the dataset

Data Cleansing

- Datasets are messy, messy data can give wrong insights (Martin Goodson's story*)
- Cleansing/Cleaning data "find and remove or correct data that detracts from the quality, and thus the usability, of data. The goal of data cleansing is to achieve consistent, complete, accurate, and uniform data"**

*https://www.martingoodson.com/ten-ways-your-data-project-is-going-to-fail/

** https://www.informatica.com/au/services-and-training/glossary-of-terms/data-cleansing-definition.html#fbid=eohTNIdjKaj

Data Cleansing

- Dealing with Missing Data
- Removing Unnecessary Data (rows, or columns)
- Normalizing/Formatting data

Dealing with Missing Data

- One of the most common problems is missing data. This could be because it was never filled out properly, the data wasn’t available, or there was a computing error. Whatever the reason, if we leave the blank values in there, it will cause errors in analysis later on. There are few ways to deal with missing data:
  - Add in a default value for the missing data
  - Get rid of (delete) the rows that have missing data
  - Get rid of (delete) the columns that have a high incidence of missing data

***http://www.developintelligence.com/blog/2017/08/data-cleaning-pandas-python/**

Dealing with Missing Data (cont’d)

- In accordance to your dataset and the task required for analysis you need to decide if you are going to use a default value to replace the missing data (NaN, 0, NA, N/A, None)
  - Don’t forget the Properties of NaN if you are doing calculations
  - Category based analysis may need special consideration
  - Some numerical analysis may benefit from default values like average, mean, median
- In Python/Pandas you can use .fillna() for filling NaN fields

Example:

Data.fillna(‘No Value’)

Removing Unnecessary Data

- Some times you don’t need all the data in the tables so it might help you achieve better performance if you remove the irrelevant data.
- Some columns or rows might be useless for you in the analysis due to having many missing values and replacing them with default values would produce wrong insights.
- Python has a very good function Drop() to help you with this

Removing Unnecessary Data Example

- Dropping Columns with all NaN values

Example: data.dropna(axis=1, how=’all’)

- Dropping Raws with all NaN values

Example: data.dropna(axis=0, how=’all’) # you don’t need to have axis=0

- Dropping Multiple Columns

to_drop = [Column1’, Column6’,Column12’, Column13’, ' Column14 ',' Column16’, Column17 ‘, 'Column18’]

data.drop(to_drop, inplace=True, axis=1)

*http://www.developintelligence.com/blog/2017/08/data-cleaning-pandas-python/

** https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.fillna.html

Normalizing/ Formatting data

- Data read from source may not have the correct format (e.g., reading integer as a string)
- Some strings in the data have spacing which might not play well with your analysis at some point.
- The date/time format may not appropriate for your analysis
- Some times the data is generated by a computer program, so it probably has some computer-generated column names, too. Those can be hard to read and understand while working.

Normalizing/ Formatting data Examples

- Example1 (change data type on read):

df = pd.read_csv(‘mydata.csv’, dtype={‘Integer_Column’: int})

- Example2 (change data type in dataframe)

df['column'] = df['column'].to_numeric()

df['column'] = df['column'].astype(str)

- Example3 (Spacing within the values):

data[‘Column_with_spacing’].str.strip()

Normalizing/ Formatting data Examples

- Example4 (unnecessary time item in the date field):

df[‘MonthYear’] = pd.to_datetime(df[‘MonthYear’])

df[‘MonthYear’] = df[‘MonthYear’].apply(lambda x: x.date())

- Example5 (rename columns)

 data = data.rename(columns = {‘Bad_Name1’:Better_Name1’, ‘Bad_Name2’:’Better_name2’})

Manipulating the data

- Merging Data
- Applying a function to data
- Pivot tables
- Change the index of a dataframe
- Groupby

Merging Data

- Sometimes in order to have complete dataset you need to Concatenate two datasets

Example:

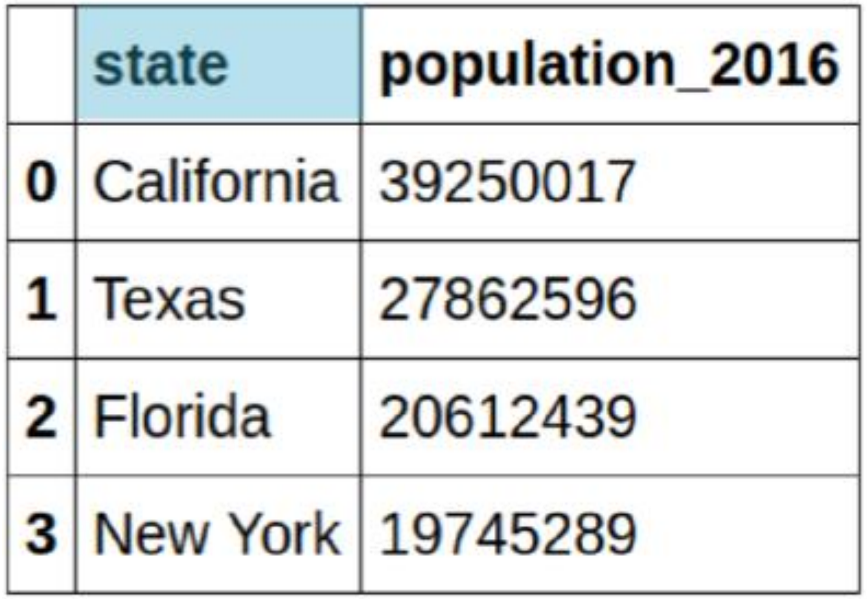
Dataset1=pd.read_csv(‘datasets/project1/dataset1.csv’)

Dataset2=pd.read_csv(‘datasets/project1/dataset2.csv’)

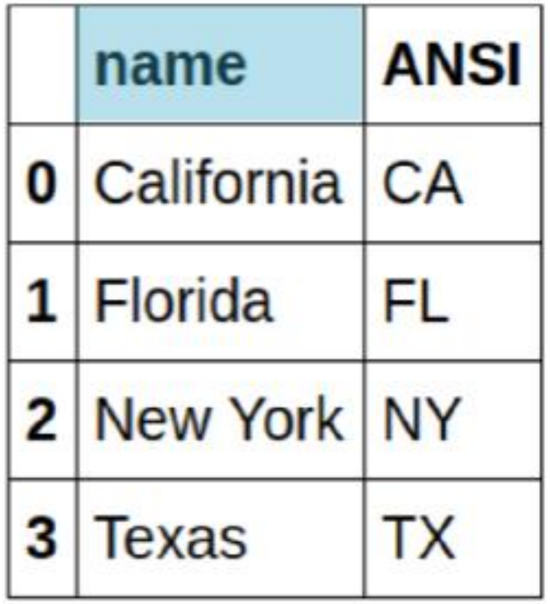
Full_data=pd.concat[Dataset1, Dataset2] axis=0, ignore_index=True)

Merging Data (Cont'd)

- Sometimes in order to have complete dataset you need to merge two Dataframes



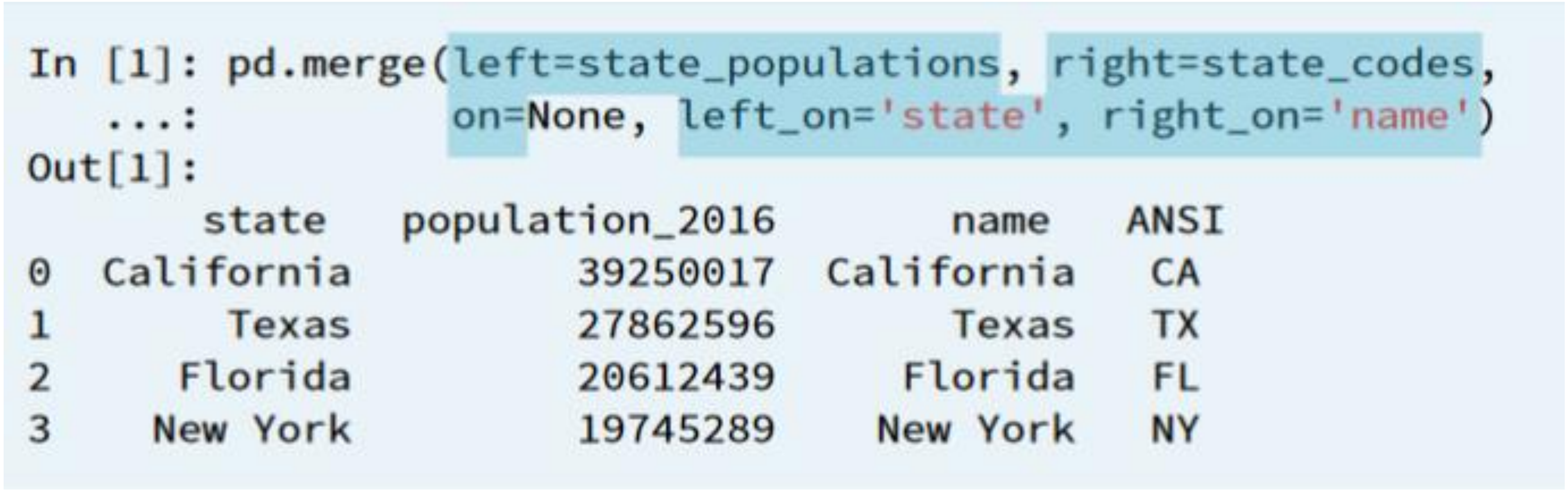
	state	population_2016
0	California	39250017
1	Texas	27862596
2	Florida	20612439
3	New York	19745289



	name	ANSI
0	California	CA
1	Florida	FL
2	New York	NY
3	Texas	TX

https://s3.amazonaws.com/assets.datacamp.com/production/course_3639/slides/ch3_slides.pdf

Merging Data (Cont'd)



```
In [1]: pd.merge(left=state_populations, right=state_codes,
   ...:              on=None, left_on='state', right_on='name')
Out[1]:
        state   population_2016        name   ANSI
0  California          39250017  California     CA
1       Texas          27862596       Texas     TX
2     Florida          20612439     Florida     FL
3    New York          19745289    New York     NY
```

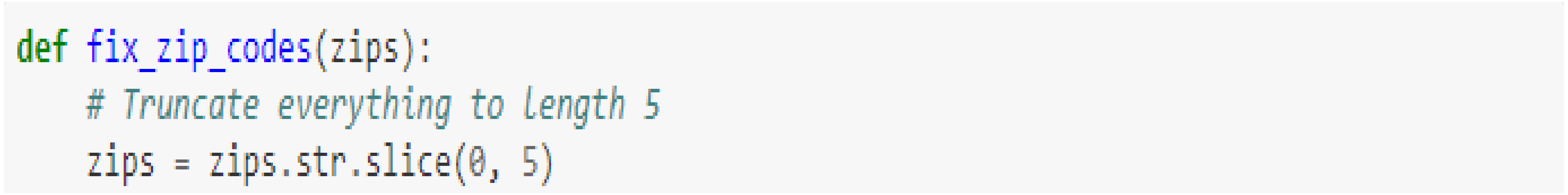
Applying a function to the entire dataset

- Sometimes You need to apply a function on the level of the entire dataset (e.g., removing, adding, averaging)

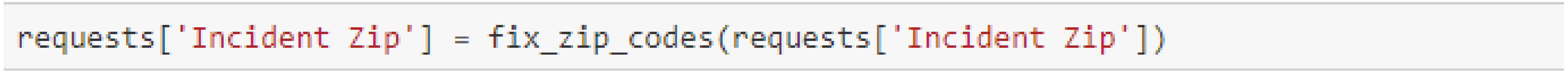
```
def cleaning_function(row_data):
        # Computation steps
        # Computation steps
df.apply(cleaning_function, axis=1)
```

Applying a Function to Columns

- Sometimes You need to apply a function on the level of Columns



```
def fix_zip_codes(zips):
    # Truncate everything to length 5
    zips = zips.str.slice(0, 5)
```



```
requests['Incident Zip'] = fix_zip_codes(requests['Incident Zip'])
```

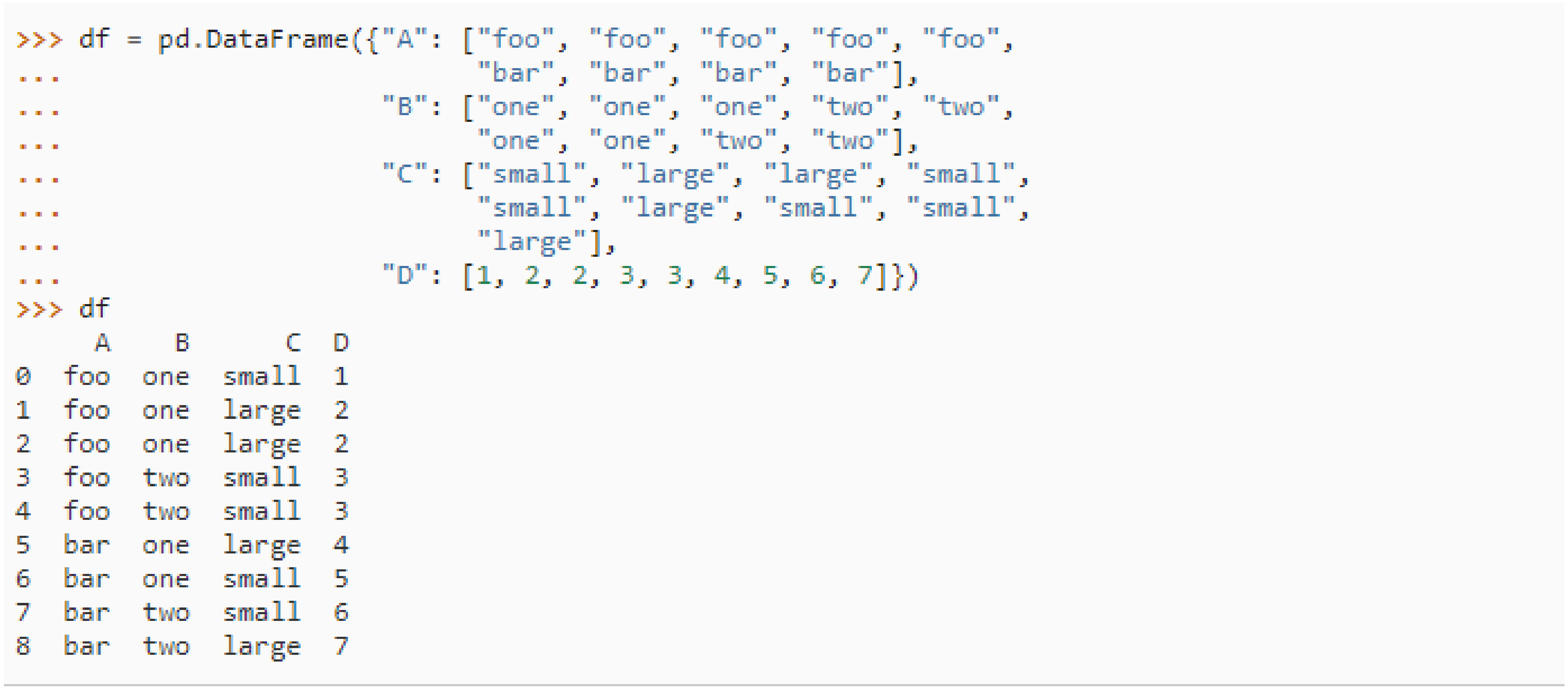
Pivot Tables

- Summary tables
- Introduce new columns from calculations
- Table can have multiple Indexes
- Excel is famous for it

*http://www.developintelligence.com/blog/2017/08/data-cleaning-pandas-python/

** https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.fillna.html

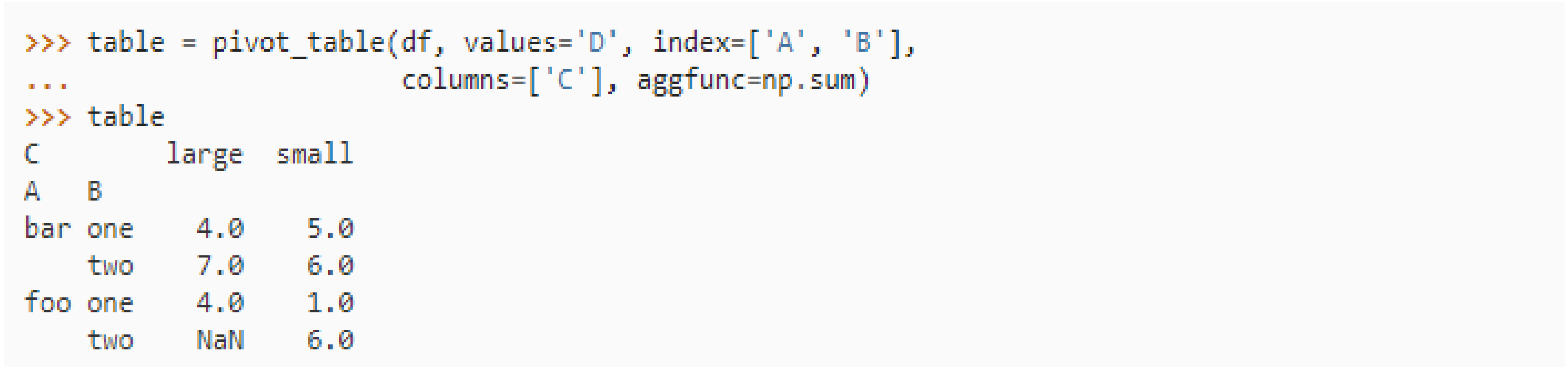
Pivot Table Example



```
>>> df = pd.DataFrame({"A": ["foo", "foo", "foo", "foo", "foo",
...                          "bar", "bar", "bar", "bar"],
...                    "B": ["one", "one", "one", "two", "two",
...                          "one", "one", "two", "two"],
...                    "C": ["small", "large", "large", "small",
...                          "small", "large", "small", "small",
...                          "large"],
...                    "D": [1, 2, 2, 3, 3, 4, 5, 6, 7]})
>>> df
     A    B      C  D
0  foo  one  small  1
1  foo  one  large  2
2  foo  one  large  2
3  foo  two  small  3
4  foo  two  small  3
5  bar  one  large  4
6  bar  one  small  5
7  bar  two  small  6
8  bar  two  large  7
```

http://pbpython.com/pandas-pivot-table-explained.html

Pivot Table Example



```
>>> table = pivot_table(df, values='D', index=['A', 'B'],
...                     columns=['C'], aggfunc=np.sum)
>>> table
C        large  small
A   B
bar one    4.0    5.0
    two    7.0    6.0
foo one    4.0    1.0
    two    NaN    6.0
```

Groupby

- Groupby splits the data into different groups depending on a variable of your choice.
- The output from a groupby and aggregation operation is it a Pandas Series or a Pandas Dataframes?
  - As a rule of thumb, if you calculate more than one column of results, your result will be a Dataframe. For a single column of results, the agg function, by default, will produce a Series.

Groupby Example

- If our dataset is tweets extracted from Twitter and we want to group all the tweets by the username and count the number of tweets each user has

```
Our_grouped_tweets= df.groupby(‘username’) [‘tweets’].count()
```

Indexing the Dataframe

- Sometimes it is helpful to use a uniquely valued identifying field of the data as its index
  - How to check uniqueness? (df[‘Unique_column'].is_unique)
  - How to set the index? (df = df.set_index(' Unique_column’))
  - Is it necessary to have unique vales in column? No, but it will affect the performance
- Pandas supports three types of Multi-axes indexing:

.loc()	Label based
.iloc()	Integer based
.ix()	Both Label and Integer based

Sorting Data

- Sometimes it is required to sort the data according to one or multiple columns.
- Pandas allow this using the function .sort_values()

Example:

```
df = pd.DataFrame({'col1' : ['A', 'A', 'B', np.nan, 'D', 'C'],'col2' : [2, 1, 9, 8, 7, 4], 'col3': [0, 1, 9, 4, 2, 3]})
```

```
df.sort_values(by=['col1'])
```

```
   col1 col2 col3
0   A    2    0
1   A    1    1
2   B    9    9
5   C    4    3
4   D    7    2
3   NaN  8    4
```

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.sort_values.html

Useful Read

- Python for Data Analysis, Wes McKinney
- https://www.altexsoft.com/blog/datascience/preparing-your-dataset-for-machine-learning-8-basic-techniques-that-make-your-data-better/
- https://pandas.pydata.org/pandas-docs/stable/tutorials.html
- https://www.analyticsvidhya.com/blog/2016/01/12-pandas-techniques-python-data-manipulation/
- https://www.dataquest.io/blog/machine-learning-preparing-data/